IPEA
EPO
D-80298 Munich
Germany

1 July 2004

Dear Sirs

PCT/GB2003/02820
Our ref: Strand

Thank you for your Written Opinion dated 1 March 2004. The Opinion is an auto-generated Opinion which states in essence that the invention as claimed lacks novelty and/or inventive step in light of the art cited in the Search Report.

The Search Report cites 3 category X documents against the independent claims:

**D1 TSAI J-Y ET AL**

**D2 MAHLKE S A ET AL**

**D3 AUGUST D I ET AL**

In light of the citations, the applicant files replacement pages as follows:

Replacement claims pages 39 - 42 to replace pages 39 - 42 as originally filed.

Amended Claim 1 of the present application reads:

1. A method of instruction execution within a microprocessor whereby:

(a) a sequence of operations are divided into individual strands;

(b) the strands are numbered to provide an implicit logical time ordering;

Origin Limited
Twisden Works, Twisden Road, London NW5 1DN Tel: +44 (0)20 7424 1950 Fax: +44 (0)20 7209 0643
E-mail: peter.langley@origin.co.uk www.origin.co.uk Registration no. 2211999

(c) certain operations from different strands may be executed out-of-order with respect to their original sequential order; and

(d) each strand has a predication status that determines whether certain operations from the strand should be completed.

This places the subject matter of original Claim 23 (underlined) into Claim 1. None of the cited art discloses the idea of numbering strands to provide an implicit logical time ordering. The use of an ordering embodied within the strand (predicate) numbering allows logically later strands to be squashed without the need for exception tags to be passed between operations. This is advantageous since at the point a speculative load is issued it is not known whether it will have to be aborted. It is only when all logically intervening stores, that may be aliased, have been issued that the final exception status of the earlier load can be determined. The cited methods require the use of an explicit data speculative check instruction. Such a check can only be performed after all machine state changing operations have reached a point consistent with the block in which the check is issued, so that a valid recovery operation can be initiated. The method described here allows the aliased store to automatically squash all later operations on the basis of the logical time ordering of the strands alone, without the need for any tags to be propagated. Such a combined store and data speculative check can be placed without any additional restrictions.

**D1 TSAI J-Y ET AL (Enhancing Multiple-Path Speculative Execution with Predicate Window Shifting)**

This describes an enhanced scheme for predication across multiple paths with an optimized tag attached to each instruction. This tag determines the set of predicates that must be either true or false for the instruction to be executed. A disadvantage of prior art techniques is that as the number of predicates in a code region increases the size of the predicate section of the instruction and significantly lowers code density. This approach uses a sliding window of predicates allowing reasonable scheduling freedom whilst retaining a short predicate tag width.

This citation uses a significantly different implementation from that described in the present application. The described mechanism uses a single predicate number for a conditional instruction. The status of this predicate is fully evaluated to determine whether the associated strand should be executed, even if it is within a nested conditional block. This representation does not suffer from the same predicate tag width expansion associated with the core technique described in this citation. Furthermore, the predicate numbers have no implicit time ordering as required by amended Claim 1.

**D2 MAHLKE S A ET AL (Effective Compiler Support for Predicated Execution Using the Hyperblock)**

This is an early paper that first described the hyperblock (or region) representation of code to reveal greater parallelism for predication and speculation. This discloses a basic predicate mechanism. It does not describe data speculation and recovery. These are key requirements to reveal greater levels of parallelism in the presence of ambiguous load/store addressing and is the primary motivation for the strand mechanism disclosed in the application. There is no predicate time ordering described in this citation, as required by amended Claim 1.

**D3 AUGUST D I ET AL (Integrated Predicate and Speculative Execution in the IMPACT EPIC architecture)**

This paper describes a significant improvement upon earlier techniques for the recovery from incorrect data speculations. Prior art techniques required the use of special compensation code to recover from incorrect speculations in particular instances. This results in a significant increase in code size. This citation discusses an inline recovery technique whereby the original code can be re-executed to recover from speculation failures. The need for such an inline recovery technique is also one of the primary motivators of the present application. The cited version uses exception tags included in the dataflow between instructions and the use of explicit data speculative check operations to cause a conditional recovery at certain points of execution. The technique of the present invention uses an alternative approach that uses an ordering of the strand (or predicate) numbers to indicate the original sequential ordering of operations. As noted above, this allows data speculations to squash later (higher numbered) strands

automatically without the need to propagate tag values or schedule explicit data speculative check operations.

In the light of the above amendments and arguments, reconsideration is respectfully requested. Should the examiner require further clarification, a second Written Opinion is requested.

Yours faithfully,

Peter Langley